



ESCUELA POLITÉCNICA NACIONAL  
ESCUELA DE FORMACIÓN DE TECNÓLOGOS  
DISEÑO DE INTERFACES



Nombre: Changoluiza Joseph

Fecha: 16/12/2024

## Consulta

### Que son los Props en React

En **React**, los **Props** (abreviatura de *properties*) son argumentos que se pasan de un componente padre a un componente hijo. Los props permiten que los componentes sean reutilizables y dinámicos, ya que se pueden enviar diferentes datos a los componentes y estos los renderizan de acuerdo a esos valores.

#### Características de los Props:

1. **Unidireccionales:** Los props fluyen en una sola dirección, del componente padre hacia el componente hijo. Es decir, los hijos no pueden modificar directamente los props que reciben.
2. **Inmutables:** Una vez que un componente recibe props, estos **no deben modificarse** dentro del componente hijo.
3. **Pasan datos dinámicos:** Puedes enviar variables, objetos, funciones y otros valores como props.
4. **Facilitan la comunicación:** Ayudan a comunicar información entre componentes.

### Cuáles son las diversas formas de dar estilos a un componente en React

En **React**, hay varias formas de aplicar estilos a un componente. Cada una tiene ventajas y desventajas dependiendo del caso de uso.

#### 1. CSS Externo (Archivos CSS)

Puedes crear un archivo .css y enlazarlo en el componente. Es la forma más tradicional y sencilla.

### Ejemplo:

1. Archivo styles.css:

css

Copiar código

```
.titulo {  
  color: blue;  
  font-size: 24px;  
}
```

2. Archivo Componente.js:

jsx

Copiar código

```
import React from "react";  
import "./styles.css"; // Importamos el archivo CSS  
  
function Componente() {  
  return <h1 className="titulo">Hola Mundo</h1>;  
}  
  
export default Componente;
```

---

## 2. Estilos Inline

Puedes aplicar estilos directamente en línea utilizando la propiedad style en formato de objeto.

### Ejemplo:

jsx

Copiar código

```
function Componente() {  
  const estilo = {
```

```
    color: "red",
    fontSize: "20px",
    backgroundColor: "yellow",
  };

  return <h1 style={estilo}>Hola Mundo</h1>;
}
```

export default Componente;

**Nota:**

- Los nombres de las propiedades CSS deben estar en **camelCase** (por ejemplo, backgroundColor en lugar de background-color).
- No es la opción más recomendada si los estilos son complejos, ya que puede resultar difícil de mantener.

---

### 3. CSS Modules

Los módulos CSS permiten encapsular estilos, evitando conflictos de nombres entre clases.

**Ejemplo:**

1. Archivo Componente.module.css:

css

Copiar código

```
.titulo {
  color: green;
  font-weight: bold;
}
```

2. Archivo Componente.js:

jsx

Copiar código

```
import React from "react";  
  
import styles from "./Componente.module.css";  
  
function Componente() {  
  return <h1 className={styles.titulo}>Hola Mundo</h1>;  
}
```

export default Componente;

### **Ventaja:**

Los estilos definidos en un módulo CSS son **locales al componente**, lo que evita conflictos entre clases.

---

## **4. Styled-Components (CSS-in-JS)**

styled-components es una biblioteca popular que permite escribir estilos directamente en JavaScript usando **template literals**.

### **Instalación:**

bash

Copiar código

```
npm install styled-components
```

### **Ejemplo:**

jsx

Copiar código

```
import React from "react";  
  
import styled from "styled-components";
```

```
const Titulo = styled.h1`
```

```
  color: purple;
```

```
font-size: 28px;
text-align: center;
`;
```

```
function Componente() {
  return <Titulo>Hola Mundo</Titulo>;
}
```

```
export default Componente;
```

### **Ventajas:**

- Los estilos son dinámicos y pueden depender de **props**.
- Evita conflictos globales de CSS.

---

## **5. CSS-in-JS con Emotion**

**Emotion** es otra biblioteca popular para escribir CSS en JavaScript.

### **Instalación:**

bash

Copiar código

```
npm install @emotion/react @emotion/styled
```

### **Ejemplo:**

jsx

Copiar código

```
/** @jsxImportSource @emotion/react */
```

```
import { css } from "@emotion/react";
```

```
function Componente() {
  const estilo = css`
    color: orange;
```

```
font-size: 24px;
text-decoration: underline;
`;

return <h1 css={estilo}>Hola Mundo</h1>;
}

export default Componente;
```

---

## 6. Frameworks de UI (Tailwind, Bootstrap, Material UI)

Puedes utilizar frameworks o bibliotecas de componentes predefinidos como **Tailwind CSS**, **Bootstrap**, o **Material-UI** para dar estilos rápidos.

### Ejemplo con Tailwind CSS:

```
jsx
Copiar código
function Componente() {
  return <h1 className="text-2xl text-blue-500">Hola Mundo</h1>;
}
```

### Ventaja:

Proporcionan estilos listos para usar, acelerando el desarrollo de interfaces.

---

## 7. Sass o Less (Preprocesadores CSS)

Puedes usar preprocesadores como **Sass** o **Less** para escribir estilos más avanzados con variables, anidamiento y funciones.

### Instalación de Sass:

```
bash
Copiar código
npm install sass
```

## Ejemplo:

### 1. Archivo styles.scss:

scss

Copiar código

```
$color-principal: blue;
```

```
.titulo {
```

```
  color: $color-principal;
```

```
  font-size: 20px;
```

```
  &:hover {
```

```
    color: darkblue;
```

```
  }
```

```
}
```

### 2. Archivo Componente.js:

jsx

Copiar código

```
import React from "react";
```

```
import "./styles.scss";
```

```
function Componente() {
```

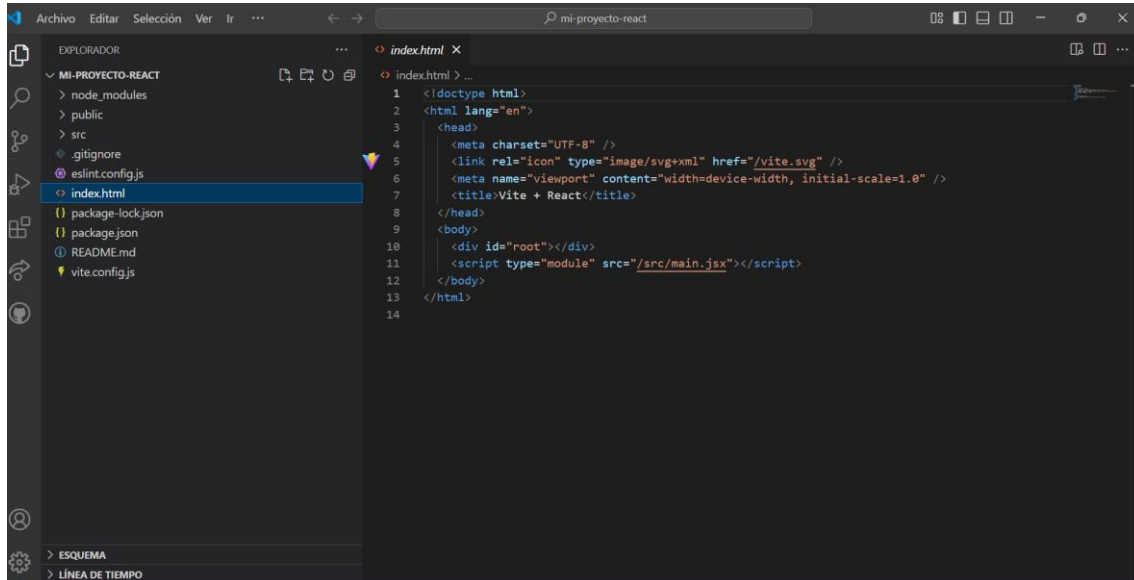
```
  return <h1 className="titulo">Hola Mundo</h1>;
```

```
}
```

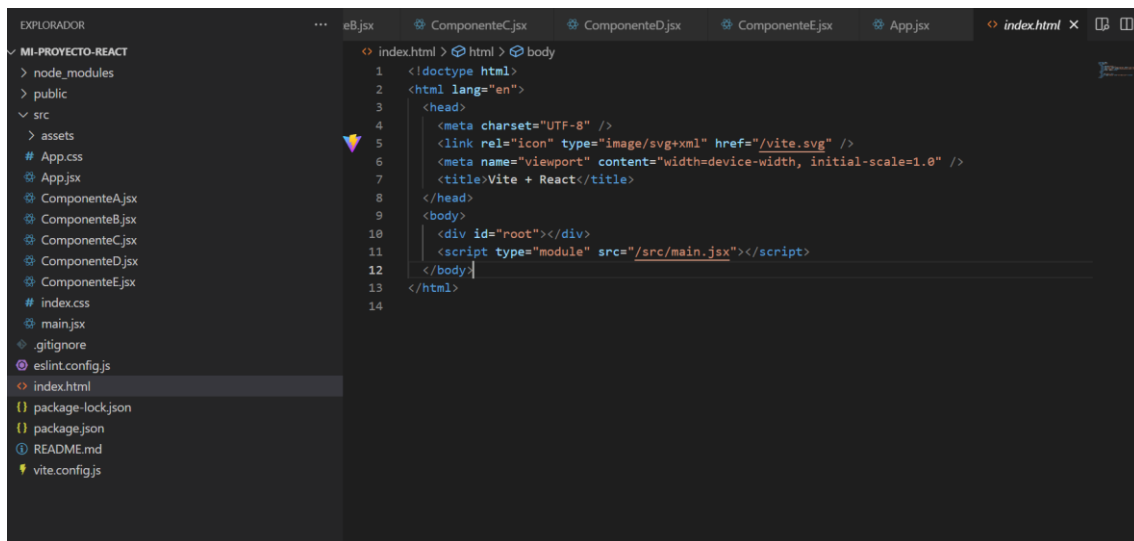
```
export default Componente;
```

## 2. Realiza la creación de

- Un proyecto en React utilizando Vite
- Crea 5 componentes y luego invócalos en el componente principal.



```
1 <!doctype html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8" />
5   <link rel="icon" type="image/svg+xml" href="/vite.svg" />
6   <meta name="viewport" content="width=device-width, initial-scale=1.0" />
7   <title>Vite + React</title>
8 </head>
9 <body>
10   <div id="root"></div>
11   <script type="module" src="/src/main.jsx"></script>
12 </body>
13 </html>
14
```



```
1 <!doctype html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8" />
5   <link rel="icon" type="image/svg+xml" href="/vite.svg" />
6   <meta name="viewport" content="width=device-width, initial-scale=1.0" />
7   <title>Vite + React</title>
8 </head>
9 <body>
10   <div id="root"></div>
11   <script type="module" src="/src/main.jsx"></script>
12 </body>
13 </html>
14
```



